Applications of Discrete Dynamical Systems with Mathematica (Study of Mathematical Software and Its Effective Use for Mathematics Education)

AUTHOR(S):

Ufuktepe, Unal; Kapcak, Sinan

数理解析研究所講究録
第1909巻 2014年 207-216

RESEARCH ARTICLE

Applications of Discrete Dynamical Systems with Mathematica

Ünal Ufuktepe*, Sinan Kapçak

İzmir University of Economics, Department of Mathematics, Izmir, Turkey

(*Received 00 Month 200x; final version received 00 Month 200x*)

Mathematica is extremely popular with a wide range of researchers from all sorts of disciplines. It is a symbolic, numerical and graphical manipulation package.

In this paper we provide an introduction to the theory of discrete dynamical systems with the aid of the Mathematica for both senior undergraduates and graduate students.

Mathematica applications cover the stability of the one dimensional system, the Cobweb diagram for one dimensional system, the time series diagram, the phase plane diagrams for two-dimensional systems. Applications are taken from biomathematics subjects: prey-predator models, host-parasitoid models, population dynamics; and modeling the populations of interacting species, bifurcation, and basin of attractions are given with codes and examples.

Keywords: discrete dynamical systems; mathematica

1. Introduction

Mathematica is extremely popular with a wide range of researchers from all sorts of disciplines. It is a symbolic, numerical and graphical manipulation package.

This paper is both an survey on theory and techniques of discrete dynamical systems by using of the software Mathematica. This paper is intended for students of mathematics, life sciences,economics, and engineering. The theory of discrete dynamical systems developed greatly during the last twenty five years of the twentieth century. Applications of difference equations also experienced enormous growth in many areas,for example in Biology. In 1985, the software program Phaser by H.Kocak appeared and made a great impact. Another important software is Dynamics, by J.Yorke's group, which appeared in 1994. Dynamics is a program written in C that, in addition to plotting trajectories, has other capabilities such as calculating of Lyapunov exponents, plotting bifurcation diagrams, and finding basins of attraction. Recent advances in the technology of Computer Algebra Systems (CAS) allow the use of symbolic calculation to study difference equations. For example, linearized stability analysis of systems with parameters, calculation of invariants, finding Lyapunov functions (based on invariants), finding symbolic periodic solutions, can all be treated with a CAS. They developed the Mathematica based package Dynamica as a collection of tools for use in the study of discrete dynamical systems and difference equations[2] .

We provide an introduction to the theory of discrete dynamical systems with the aid of the Mathematica with codes for both senior undergraduates and graduate students.

*Corresponding author. Email: unal.ufuktepe@ieu.edu.tr

2. One Dimensional Models

A population is defined as a group of individuals of the same species within a limited area. Mathematical models are used to predict the size or density (population size per unit area) of a population at any time in the future. They are also used to check the biological assumptions that are made to produce the model. Let X_t be the size (density) of a population at time t, and X_{t+1} be the size (density) of this population at the next time interval or generation. Then X_{t+1} is related to X_t by a function f which may be written in the form $X_{t+1} = f(X_t)$.

2.1. *Stability of Equilibrium points*

A point x^* is said to be a fixed point (or equilibrium point) of a map f if $f(x^*) = x^*$. It is important to develop qualitative or graphical methods to determine the behavior of the orbits $(\{x_0, f(@_0), f(f(x_0)), ...\})$ near fixed points. Such a program of investigation is called stability theory. x^* is said to be stable if for any $\varepsilon > 0$ there exists $\delta > 0$ such that for all $x_0 \in I$ with $|x_0 - x^*| < \delta$ we have $|f^n(x_0) - x^*| < \varepsilon$ for all n. Otherwise, It is called unstable.

Theorem:(For hyperbolic fixed points) If $|f'(x^*)| < 1$ then x^* is stable. If $|f'(x^*)| > 1$ then x^* is unstable

Theorem: (For nonhyperbolic fixed points)

(1) If $f'(x^*) = 1$ (f', f'', f''' are continuous at x^*)

   a) If $f''(x^*) \neq 0$, then x^* is unstable (semistable)
   b) If $f''(x^*) = 0$ and $f'''(x^*) > 0$, then x^* is unstable
   c) If $f''(x^*) = 0$ and $f'''(x^*) < 0$, then x^* is asymptotically stable

(2) If $f'(x^*) = -1$, (f', f'', f''' are continuous at x^*) :

   a) If $Sf(x^*) < 0$ then x^* is asymptotically stable where $(Sf(x) = \frac{f'''(x)}{f'(x)} - \frac{3}{2}(\frac{f''(x)}{f'(x)})^2)$
   b) If $Sf(x^*) > 0$ then x^* is unstable [1]

By the following code when the user enters the function $f(t)$ as $OneDimStability[f[t], t]$ then he/she will get the type of fixed point(s) and whether it is stable or not.

```
OneDimStability[F1_, x1_] := Module[{},
    OneDimStabilityy[F_, x_, p_] := Module[{},
        If[(F /. x -> p) != p, Print[p, " is not a fixed point!"],
     If[Abs[(D[F, {x, 1}] /. x -> p)] < 1,
      Print[p, ": Hyperbolic, Stable"]];
     If[Abs[(D[F, {x, 1}] /. x -> p)] > 1,
      Print[p, ": Hyperbolic, Unstable"]];
     Schwarzian[f_, xx_] :=
     D[f, {xx, 3}]/(D[f, xx]) - (3/2) ((D[f, {xx, 2}])/(D[f, xx]))^2;
     If[(D[F, {x, 1}] /. x -> p) == 1,
      If[(D[F, {x, 2}] /. x -> p) != 0,
       Print[p, ": Nonhyperbolic, Unstable (Semistable)"],
       Which[(D[F, {x, 3}] /. x -> p) > 0,
```

```
      Print[p,
       ": Nonhyperbolic, Unstable"], (D[F, {x, 3}] /. x -> p) < 0,
      Print[p,
       ": Nonhyperbolic, Stable"], (D[F, {x, 3}] /. x -> p) = 0,
      If[(D[F, {x, 4}] /. x -> p) != 0,
       Print[p, ": Nonhyperbolic, Unstable"]]]]];

   If[(D[F, {x, 1}] /. x -> p) == -1,
    Which[(Schwarzian[F, x] /. x -> p) > 0,
     Print[p,
      ": Nonhyperbolic, Unstable"], (Schwarzian[F, x] /. x -> p) <
      0, Print[p, ": Nonhyperbolic, Stable"]]]
       ]
      ];
  aa = Solve[(F1 /. x1 -> t) == t, t];
  taa = Transpose[aa];
  sol = First[taa];
  soll = Function[Last[#]] /@ sol;
  sonlist = {};
  sonlistt =
 If[Im[soll[[#]]] == 0, Append[sonlist, soll[[#]]], sonlist] & /@
  Range[Length[soll]];
  sonlist = Union[Flatten[sonlistt]];
  First[OneDimStabilityy[F1, x1, #] & /@ sonlist]]
```

Example 2.1

```
In[1] OneDimStability[t^4 - 2 t^3 - t^2 + 3 t, t]
Out[1] -1: Hyperbolic, Unstable
0: Hyperbolic, Unstable
1: Nonhyperbolic, stable
2: Hyperbolic, Unstable
```

3. Cobweb Diagram

One of popular graphical methods to study the dynamics of first order difference equations is the cobweb diagram. To plot the cobweb-diagram, first draw the curves $y = f(x)$ and $y = x$ on the same graph. The intersections points are the fixed point(s). Then starting at x_0 which is either from left hand side of the fixed point or from right hand side , we pinpoint the value $x_1 = f(x_0)$ by drawing a vertical line through the point $(x_0, 0)$ so that it will intersect the graph of f at the point (x_0, x_1). Next we draw a horizontal line from $((x_0, x_1)$ to meet the diagonal line $y = x$ at the point (x_1, x_1). A vertical line drawn from the point (x_1, x_1) will intersect the graph of f at the point (x_1, x_2). Continuing this process, one may find $x_3, x_4, \cdots, x_t, \cdots)$ for all $t > 0$. If these sequence(s) converge to the fixed point then we say that the fixed point is stable otherwise it is unstable. In our code,

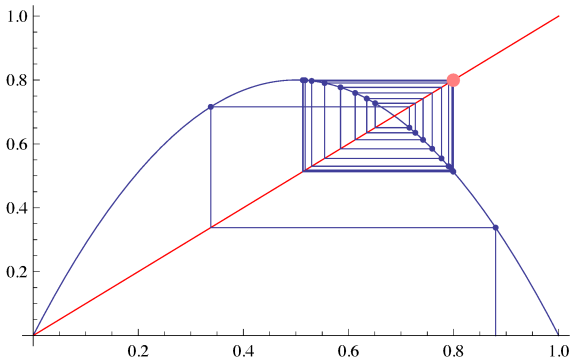
Cobweb[Function(var),var,Initial Point,interval,iteration]

when the user types the function with initial point x_0, the interval of fixed point, and the number of iteration then the user can get the Cobweb diagram. In this diagram the Pink Point shows the terminal point after n iterations.

```
Cobweb[F_, t0_] :=
 Module[{ff, diagonal, funct, list, l, ldot, ttt, sonlist},
  ff[a_] := F /. x -> a;
  diagonal = Plot[x, {x, 0, 1}, PlotStyle -> Red];
  funct = Plot[F, {x, 0, 1}];
  list = Table[{Nest[ff, t0, n], Nest[ff, t0, n + 1]}, {n, 0, 100}];
  ldot = ListPlot[list];
  sonlist = {{t0, 0}};
  For[i = 1, i < 101, i++, sonlist = Append[sonlist, list[[i]]];
   sonlist =Append[sonlist, {sonlist[[2 i]][[2]], sonlist[[2 i]][[2]]}]];
  l = ListPlot[sonlist, Joined -> True];
  ttt = Graphics[{PointSize[Large], Pink, Point[Last[sonlist]]}];
  Show[ldot, funct, diagonal, l, ttt]]
```

Example 3.1

```
In[2]: Cobweb[3.2 x (1 - x),.88,(0,2),20]
```



4. Time Series Diagram

Code

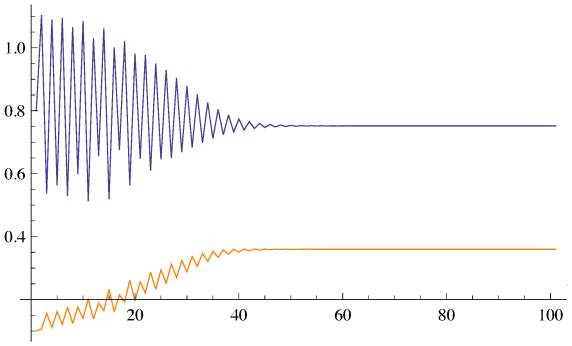
```
DDSPhasePlane1[fg_, varx_, vary_, x0_, y0_] :=
 Module[{F, G, X, Y, p0, lll},
  F[x_, y_] := fg[[1]] /. {varx -> x, vary -> y};
  G[x_, y_] := fg[[2]] /. {varx -> x, vary -> y};
  X[a_] := F[a[[1]], a[[2]]];
  Y[b_] := G[b[[1]], b[[2]]];
  p0 = {x0, y0};
  lll = NestList[{X[{#[[1]], #[[2]]}], Y[{#[[1]], #[[2]]}]} &, p0,
    100];
  l1 = ListPlot[Transpose[lll][[1]], Joined -> True, PlotRange -> All];
  l2 = ListPlot[Transpose[lll][[2]], Joined -> True,
    PlotStyle -> Orange];
  Show[l1, l2]]
```

In this code the user can get the graph of the population(s) with respect to time to see the behavior of the population whether it is stable or not. In this command when the user enters the fitness functions (for two dimensional dynamical systems), variables names, and initial values of the populations then the user can get the time

series.

Example 4.1 This example for a prey-predator model with $x_0 = 0.8, y_0 = 0.1$ initial values

```
DDSPhasePlane1[{x + 2.9 x (1 - x) - 2 x y, 1.33 x y}, x, y, .8, .1]
```



we can see the fixed points and stability of the system around the initial point in this time series graph .

5. Phase Diagram for two dimensional discrete dynamical systems

Consider the following two-dimensional discrete dynamical systems

$$x_{t+1} = f(x_t, y_t)$$

$$y_{t+1} = g(x_t, y_t)$$

The fixed point(s) of this system is the solution of the following system

```
f(x,y)=x
g(x,y)=y
```

Nicholson-Bailey Model (1935) is a model to abiological system involved two insacts, a parasitoid and its host; a parasite is free living as an adult but lays eggs in the larvae or pupae of the host. Hosts that are not parasitized give rise to their own progeny. Hosts that are successfully parasitized die but the eggs laid by the parasitoid may survive to the next generation of parasitoids.

$$N_{t+1} = rN_t f(N_t, P_t)$$
$$P_{t+1} = eN_t(1 - f(N_t, P_t))$$

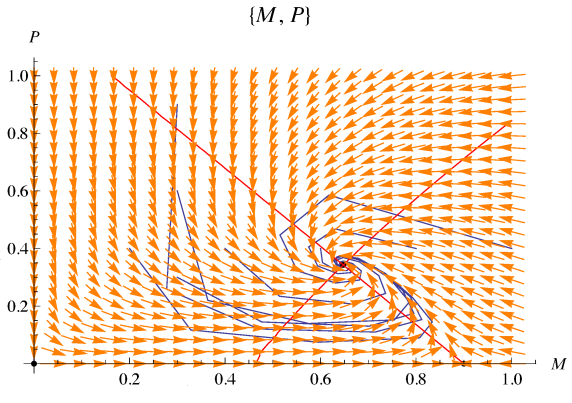
where r is the number of eggs laid by the host that survive, e is the number of eggs laid by the parasitoid on a single host that survive, f is fraction of hosts not parasitized.
Predator-Prey Models are similar to both host-parasite/parasitoid models. However, unlike the latter two systems, the predator does not live on the host. The prey serves as a food source for the predator. The following code gives the phase diagram of two dimensional systems and the orbit of the given initial point.

Code

```
DDynamicss[fg_, varx_, vary_, ss_, rr_] :=
Module[{horizontalinitials, verticalinitials, x, y, ppvector, dyn,
   gr1, gr2},PtoPvector[fg1_, varx1_, vary1_, x01_, y01_] :=
   Module[{F, G, X, Y, p0, lll, lp, ttt},
    F[x_, y_] := fg1[[1]] /. {varx1 -> x, vary1 -> y};
    G[x_, y_] := fg1[[2]] /. {varx1 -> x, vary1 -> y};
    X[a_] := F[a[[1]], a[[2]]];
    Y[b_] := G[b[[1]], b[[2]]];
    p0 = {x01, y01}; q0 = {X[p0], Y[p0]}; q0 - p0];
  ppvector =VectorPlot[PtoPvector[fg, varx, vary, x, y], {x, 0, 1}, {y, 0, 1},
  VectorPoints -> Fine, VectorScale ->
  {Automatic, Automatic, None},VectorStyle -> Orange];
  DDSPhasePlane2[fg2_, varx2_, vary2_, x02_, y02_, iterate_] :=
   Module[{F, G, X, Y, p0, lll, lp, ttt},
    F[x_, y_] := fg2[[1]] /. {varx2 -> x, vary2 -> y};
    G[x_, y_] := fg2[[2]] /. {varx2 -> x, vary2 -> y};
    X[a_] := F[a[[1]], a[[2]]];
    Y[b_] := G[b[[1]], b[[2]]];
    p0 = {x02, y02};
    lll =NestList[{X[{#[[1]], #[[2]]}], Y[{#[[1]], #[[2]]}]} &, p0,
      iterate];
    lp = ListPlot[lll, Joined -> True, AxesLabel -> {varx2, vary2},
      PlotRange -> All, PlotLabel -> {varx2, vary2}];
    Show[lp, Graphics[Point[Last[lll]]]]];
  dyn[xx0_, yy0_] := DDSPhasePlane2[fg, varx, vary, xx0, yy0, 1000];
  horizontalinitials = Table[dyn[nx, rr], {nx, 0, 1, .2}];
  verticalinitials = Table[dyn[ss, ny], {ny, 0, 1, .3}];
  gr2 = ContourPlot[{varx == fg[[1]], vary == fg[[2]]}, {varx, 0,
     1}, {vary, 0, 1}, ColorFunction -> Hue];
  Show[{horizontalinitials, verticalinitials, gr2, ppvector}]]
```

Example 5.1 This example gives the phase diagram of a host-parasitoid model and the orbit of the given $(N_0, P_0) = (0.3, 0.4)$ initial point with manifolds

```
DDynamicss[{M Exp[1.1 (1 - M/.9) - .9 P],
M (1 - Exp[-2.2 P])}, M, P, .3, .4]
```



6. Bifurcation Diagram

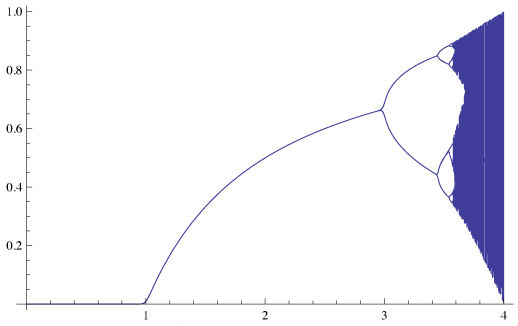
In general, the term bifurcation refers to the phenomenon of a system exhibiting new dynamical behavior as the parameter (r) is varied in the one dimensional system $x_{t+1} = f(x_t, r)$. Bifurcation is calcified withe respect to the following rules:
Saddle node bifurcation if $\frac{\partial f(x^*,r^*)}{\partial x} = 1, \frac{\partial f(x^*,r^*)}{\partial r} \neq 0$, and $\frac{\partial^2 f(x^*,r^*)}{\partial x^2} \neq 0$
Pitchfork bifurcation if $\frac{\partial f(x^*,r^*)}{\partial x} = 1, \frac{\partial f(x^*,r^*)}{\partial r} = 0$, and $\frac{\partial^2 f(x^*,r^*)}{\partial x^2} = 0$
Transcritical bifurcation if $\frac{\partial f(x^*,r^*)}{\partial x} = 1, \frac{\partial f(x^*,r^*)}{\partial r} = 0$, and $\frac{\partial^2 f(x^*,r^*)}{\partial x^2} \neq 0$
Period doubling bifurcation if $\frac{\partial f(x^*,r^*)}{\partial x} = -1, \frac{\partial f(x^*,r^*)}{\partial r} \neq 0$, and $\frac{\partial^2 f(x^*,r^*)}{\partial x^2} \neq 0$ [1]

Code

```
Bif1D[f_, varx_, a_] := Module[{T, Iter, g},
  g[xx_] := f /. varx -> xx;
  Iter[k_] := Nest[g, 0.4, k];
  T := Table[Iter[n], {n, 100, 107}];
  Plot[{T}, a, AxesOrigin -> {a[[2]], 0}]]
```

Example 6.1 The following example shows the period doubling bifurcation of a Logistic model

```
Bif1D[r x (1 - x), x, {r, 0, 4}]
```



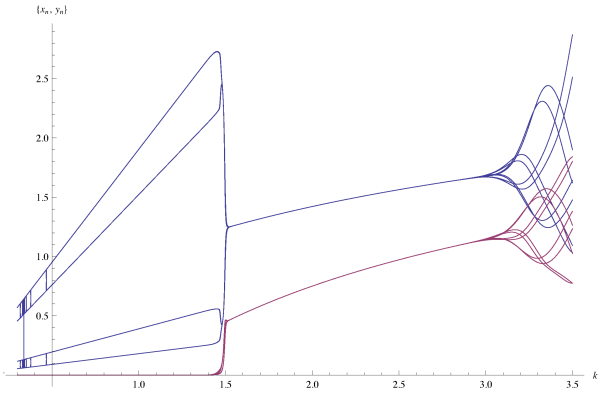
We developed the following code for two dimensional discrete dynamical systems

Code

```
Bif2D[fg_, varx_, vary_, x0_, y0_, param_, interval_] :=Module[{F, G, X, Y, p0, lll},
  F[x_, y_] := fg[[1]] /. {varx -> x, vary -> y};
  G[x_, y_] := fg[[2]] /. {varx -> x, vary -> y};
  X[a_] := F[a[[1]], a[[2]]];
  Y[b_] := G[b[[1]], b[[2]]];
  p0 = {x0, y0};
  Iterasyon[d_] :=Nest[{X[{#[[1]], #[[2]]}], Y[{#[[1]], #[[2]]}]} &, p0, d];
  Tx := Table[Iterasyon[n][[1]], {n, 80, 87}];
  Ty := Table[Iterasyon[n][[2]], {n, 80, 87}];
  Plot[{Tx, Ty}, {param, interval[[1]], interval[[2]]},PlotRange -> All,
   AxesLabel -> {param, {Subscript[x, n], Subscript[y, n]}}]]
```

Example 6.2 The following example for the host-parasitoid model

```
Bif2D[{x Exp[r (1 - x/4) - y], x (1 - Exp[-y])},
 x, y, .3, .2, r, {.3, 3}]
```



7. Basin of Attractions

Definition: Let x^* be a fixed point of a map f. Then the basin of attraction (or the stable set) $W^s(x^*)$ of x^* is defined as

$$W^s(x^*) = \{x : \lim_{n\to\infty} f^n(x) = x^*\}$$

Example 7.1 $x_{n+1} = x_n^2$: 1 and 0 are fixed points. $W^s(0) = (-1, 1)$, 1 is unstable fixed point.

We developed the following command for the basin of attraction

Code

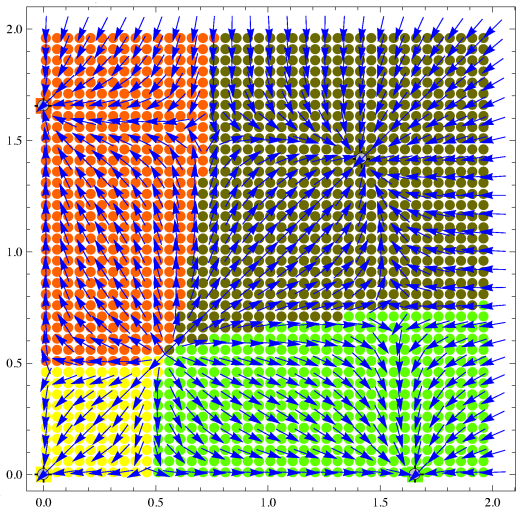
```
BasinOfAttraction[fg_, varx_, vary_, radius_, iterationnumber_,
  xintv_, yintv_] :=
 Module[{horizontalinitials, verticalinitials, x, y, ppvector, dyn,
   gr1, gr2},
  PtoPvector[fg1_, varx1_, vary1_, x01_, y01_] :=
   Module[{F, G, X, Y, p0, lll, lp, ttt},
    F[x_, y_] := fg1[[1]] /. {varx1 -> x, vary1 -> y};
    G[x_, y_] := fg1[[2]] /. {varx1 -> x, vary1 -> y};
    X[a_] := F[a[[1]], a[[2]]];
    Y[b_] := G[b[[1]], b[[2]]];
    p0 = {x01, y01}; q0 = {X[p0], Y[p0]}; q0 - p0];
  ppvector = VectorPlot[PtoPvector[fg, varx, vary, x, y],
  {x, xintv[[1]], xintv[[2]]}, {y,yintv[[1]], yintv[[2]]},
  VectorPoints -> Fine,VectorScale -> {Automatic, Automatic, None},
  VectorStyle -> Blue];
  DDSPhasePlane2[fg2_, varx2_, vary2_, x02_, y02_, iterate_] :=
  Module[{F, G, X, Y, p0, lll, lp, ttt},
    F[x_, y_] := fg2[[1]] /. {varx2 -> x, vary2 -> y};
    G[x_, y_] := fg2[[2]] /. {varx2 -> x, vary2 -> y};
    X[a_] := F[a[[1]], a[[2]]];
    Y[b_] := G[b[[1]], b[[2]]];
```

```
    p0 = {x02, y02};
    lll = NestList[{X[{#[[1]], #[[2]]}], Y[{#[[1]], #[[2]]}]} &, p0,
      iterate];
    lp = ListPlot[lll, Joined -> True, AxesLabel -> {varx2, vary2},
      PlotRange -> All, PlotLabel -> {varx2, vary2}];
    renk =If[Sqrt[(lll[[iterationnumber]][[1]] -
            lll[[iterationnumber - 1]][[
             1]])^2 + (lll[[iterationnumber]][[2]] -
            lll[[iterationnumber - 1]][[2]])^2] < radius,
      RGBColor[1/(1 + Abs[Last[lll][[1]]]),
       1/(1 + Abs[Last[lll][[2]]]), 0], Blue];
    Show[Graphics[{PointSize[Large], renk, Point[First[lll]]},
      Frame -> True],Graphics[{Locator[lll[[iterationnumber]], Background -> renk,
        Appearance -> Small]}]]];
  dyn[xx0_, yy0_] :=DDSPhasePlane2[fg, varx, vary, xx0, yy0, iterationnumber];
  verticalinitials =Table[Table[dyn[sss, ny], {ny, yintv[[1]], yintv[[2]], .05}], {sss,
      xintv[[1]], xintv[[2]], .05}];
  gr2 = ContourPlot[{varx == fg[[1]], vary == fg[[2]]}, {varx,
     xintv[[1]], xintv[[2]]}, {vary, yintv[[1]], yintv[[2]]},ColorFunction -> Hue];
  Show[{verticalinitials, ppvector}]]
```

This example for a competion model

Example 7.2

```
a = 2.116; r = .2372; b = .1606;
BasinOfAttraction[{(a x^2 + r x)/(1 + x^2 + b y), (a y^2 + r y)/(1 +
     y^2 + b x)}, x, y, .01, 200, {.01, 2}, {.01, 2}]
```



8. Conclusion

The modules we create can be used as the main tool, for the students who wish not to emphasize proofs, or as a supplement for an undergraduate course in discrete dynamical systems and difference equations. It can also be useful to graduate students and researchers studying higher order dynamics. It is hard to find the positive fixed point(s) and the Namer-Sacker bifurcation of the map in some dis-

crete dynamical systems in this case we need to either use the some numerical methods to approximate the fixed point/Namer -Sacker bifurcation. In our study find at least the smallest region of the fixed point. We have some modules for these processes for specific biological models and traced determinant condition for the higher order dynamics but because of the page limitation we could not give in this paper. In future studies we will generalize these modules for the general two dimensional systems.

References

[1] S. Elaydi, Discreté Chaos:With Applications in Science and Engineering, Chapman and Hall/CRC, Second Edition, (2008).

[2] M.R.S Kulenovic and O. Merino, Discrete Dynamical Systems and Differnce Equations with Mathematica, A CRC Press Company,2002